5 example optimization problems

I have several goals in starting this course with examples:

- It shows how optimization comes from real-world applications.
- It allows you to discover for yourself some basic theoretical/numerical ideas.
- These examples help you practice and learn some programming.[1]

The textbook, namely Griva, Nash, and Sofer, *Linear and Nonlinear Optimization*, 2nd ed., SIAM Press 2009, also provides many examples; see Chapter 1. In any case, one cannot understand theory and algorithms without some understanding of applications. All optimization experts have learned from such examples.

Each example has a name written in parentheses. I will also use this in naming my MATLAB/OCTAVE code in the solutions to Assignment # 1, e.g. `calc.m` for the first example.

Following the ideas in Chapter 2.1 in the textbook, each example identifies a *feasible set* S and an *objective function* $f(x)$, which writes it in this standard form:

$$\min_{x\in S} f(x).$$

This document does *not* address how to solve these example problems! That will be done by you; see Assignment #1 for specific expectations. When you solve one of these problems your method may be, and often will be, "brute force" and inefficient. That is just fine for now! The rest of the course will make more sense if you see some brute force approaches, before getting more elegant algorithms.

1. (`calc`) Let

$$f(x) = \left(x^2 + \sin x\right)^2 - 10\left(\cos(5x) + \frac{3}{2}x\right).$$

Compute the minimum of f on the interval $S = [0, 2]$:

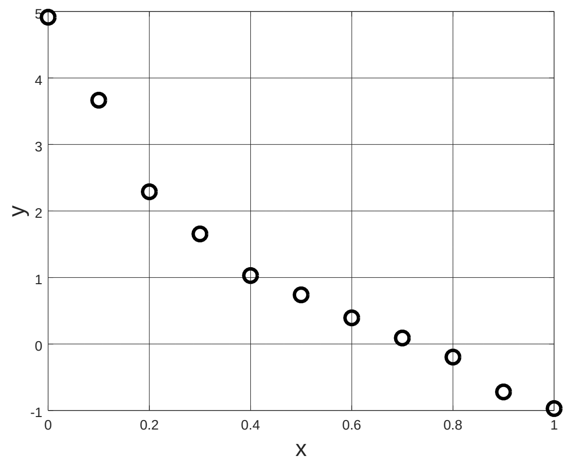
$$\min_{x\in[0,2]} f(x)$$

You saw such problems in Calculus I, but this one is a bit harder to do by hand. It benefits from computer visualization, and, because S is one-dimensional, you may easily plot $f(x)$ on S. From such a plot you can get close to the solution just by looking.

2. (`fit`) Consider the following 11 data points which are plotted below:

x	0.000	0.100	0.200	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
y	4.914	3.666	2.289	1.655	1.029	0.739	0.393	0.090	-0.197	-0.721	-0.971

[1] In this course you may use any language you want. I will provide demos, examples, and solutions in MATLAB/OCTAVE. My other recommended languages are PYTHON and JULIA.



Suppose we believe that this data can be fit by a function of the form

$$g(x) = c_1 + c_2 x + c_3 e^{-3x}.$$

Let's decide that the meaning of "fit" is that the sum of the squares of the misfits should be as small as possible. Then the problem is

$$\min_{c \in \mathbb{R}^3} f(c)$$

where we define the objective function

$$f(c) = \frac{1}{2} \sum_{j=1}^{11} \left(g(x_j) - y_j\right)^2 = \frac{1}{2} \sum_{j=1}^{11} \left(c_1 + c_2 x_j + c_3 e^{-3x_j} - y_j\right)^2.$$

(The overall factor of $1/2$ is merely a convenience when differentiating.)

Note $S = \mathbb{R}^3$, because here there are no constraints on the coefficients c_i. Also note that we are *not* finding x_j or y_j values in the minimization process! We are finding c_1, c_2, c_3. The data values (x_j, y_j) do, however, determine the objective function f.

3. (`salmon`) Ed and Vera caught 21 salmon. Of these, x_1 will be eaten fresh, which requires 2 time units per fish. Then x_2 will be vacuum-packed and frozen (3 time units per fish) and another x_3 will be smoked and vacuum-packed (4 time units per fish). Thus the total amount of processing time is $f(x) = 2x_1 + 3x_2 + 4x_3$. However, at most 2 fish can be eaten fresh before they go bad, at most 10 fish can be smoked in the smoker time available, and at least 4 fish must be smoked because they'll be mailed unfrozen to relatives. Find x_1, x_2, x_3 to minimize the total processing time.

This is a constrained minimization problem wherein x_i are numbers of fish, which must be non-negative, and the objective function is the total processing time:

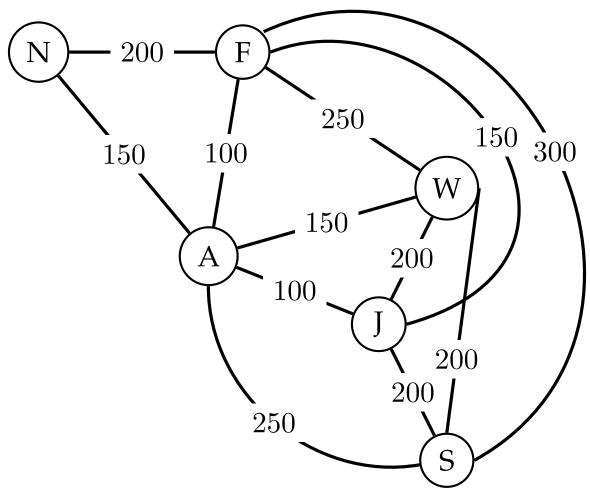
$$\min f(x) = 2x_1 + 3x_2 + 4x_3 \qquad \text{subject to} \qquad \begin{array}{c} x_1 + x_2 + x_3 = 21 \\ 0 \le x_1 \le 2 \\ 0 \le x_2 \\ 4 \le x_3 \le 10 \end{array}$$

The feasible set $S \subset \mathbb{R}^3$ includes all the constraints:

$$S = \left\{ x \in \mathbb{R}^3 \,\middle|\, x_1 + x_2 + x_3 = 21, 0 \le x_1 \le 2, 0 \le x_2, \text{ and } 4 \le x_3 \le 10 \right\}.$$

The objective function and the constraint functions (e.g. $g_1(x) = x_1 + x_2 + x_3$, $g_2(x) = x_1$, $\dots$) are linear functions, so this is a *linear programming* problem.

4. (`tsp`) Jill sells amazing widgets that help you learn math. To sell these devices she plans to visit six cities A, F, J, N, S, W by starting and ending at city F. Some cities have connecting flights and some do not; the one-way costs of the various flights are shown below in a *graph* with costs (weights) on each connection (edge). Except for starting and ending at F, it is clear that she should visit each city exactly once.



This is an example of the famous *traveling salesperson problem*. Each possible itinerary is expressible as a seven-letter string like "FWJSANF". If x denotes such a feasible string then we may define the objective function $f(x)$ to be the cost of that itinerary; thus $f(x)$ is defined using the edge weights. Even finding a feasible itinerary, a *Hamiltonian cycle*, for a big-enough graph, is generally nontrivial. One may, however, add-in all missing edges with large weights so that any itinerary x is feasible and has a well-defined cost $f(x)$.

The problem *could* be written in standard form

$$\min_S f(x).$$

where $S = \{x \,|\, x \text{ is a feasible itinerary}\}$. However, there is no easy way to describe S by inequalities and equalities as as subset of some Euclidean space $\mathbb{R}^n$ as above.

In any case, this is a *discrete optimization* problem. That is, S is a finite set of feasible itineraries. Mostly we will consider continuous optimization problems in this course, not discrete ones.

5. (`glacier`) The shape of a glacier on flat bedrock is approximately given by the solution to a constrained optimization problem. The mathematical form of this problem must express three ideas: (i) glaciers are created where snowfall exceeds melt, (ii) the glacier shape is influenced by the downhill flow of the ice, and (iii) the thickness of a glacier is a nonnegative function.

The optimal solution to the minimization problem is itself a *function* $u(x)$ giving the thickness at each location. The objective function $f[u]$ takes such a function $u(x)$ as input and produces a single real number. It is common to call such functions, which take other functions as inputs, *functionals*.

The feasible set in this specific problem is a set of functions defined on an interval:

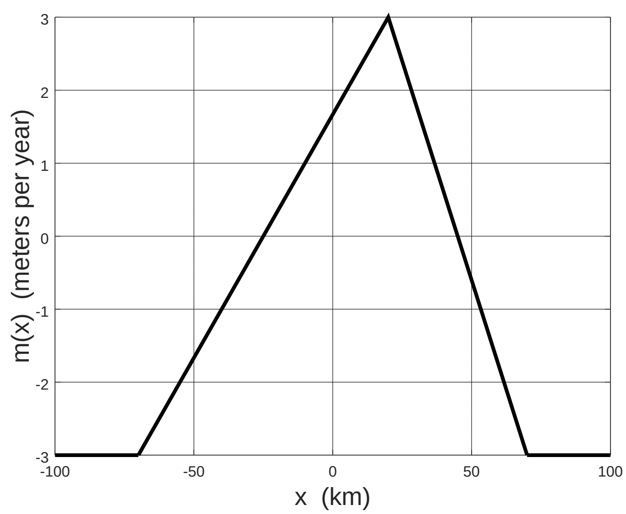
$$S = \left\{u(x) \,\middle|\, u(x) \geq 0 \text{ is a differentiable function on } [-100, 100] \text{ km}\right\}.$$

In contrast to all the above examples, an element of the feasible set S is not a finite list (vector) of numbers, it is a function itself. Therefore this *calculus of variations* problem is infinite-dimensional.

In this specific problem I will make up a source function which defines the rate of total snowfall or melt in a year, the *mass balance* in glaciologist language:

$$m(x) = \begin{cases} -3 + \frac{6}{90}(x+70), & -70 \le x \le 20, \\ 3 - \frac{6}{50}(x-20), & 20 \le x \le 70, \\ -3, & \text{otherwise.} \end{cases}$$

The units of $m(x)$ are meters per year. This function is graphed below. Note that it is only snowing where $m(x)$ is positive; everywhere else it is melting.



The objective functional is an integral defined using the data $m(x)$:

$$f[u] = \int_{-100}^{100} \frac{\mu}{4} \left(u'(x)\right)^4 - m(x)u(x)\, dx$$

Based on other physical constants related to the slow flow of ice (not shown) we set $\mu = 5 \times 10^{-14}$. The glacier shape is derived from the solution of the problem

$$\min_S f[u].$$

Once $u(x)$ is computed we raise it to a power to get the actual ice thickness $H(x)$, measured in meters, of the glacier:

$$H(x) = u(x)^{3/8}.$$

Because this problem uses an infinite-dimensional feasible set S, computer solutions require *discretization*. The objective functional can only be evaluated approximately. (In fact, storing arbitrary functions of x on an interval is not possible.) The easiest way to discretize is to put a grid on the interval $I = [-100, 100]$ and only consider functions which are piecewise-linear between the points of this grid. In that case the derivative $u'(x)$ in the integral for $f[u]$ is computed by the difference quotient which gives the slope of the line segment between the points. Evidently, we want the grid to be as fine as practical given our tools, that is, given the available optimization algorithms and computer resources.